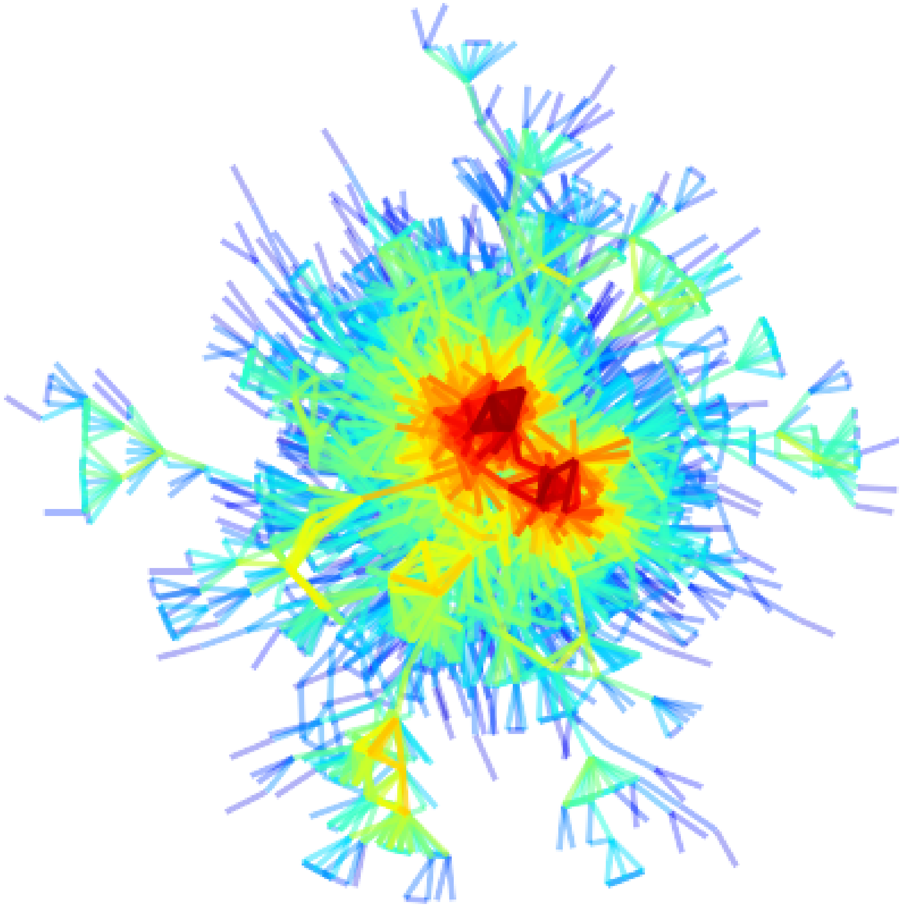
IMDb Collaboration Graph

building and analisys of properties



Mario Paoli
mariozz@hotmail.it

Luca Favretto
lucafavretto@gmail.com

Dipartimento di Ingegneria Informatica, Automatica e Gestionale
Antonio Ruberti
Sapienza Università di Roma

IMDb®

Earth's Biggest Movie Database™

Internet Movie Database (IMDb) is an online database of information related to movies, television shows, actors, production crew personnel and fictional characters featured in visual entertainment media. It is one of the most popular online entertainment destinations, with over 100 million unique users each month and a solid and rapidly growing mobile presence.[Wikipedia]

IMDb make its database downloadable in different formats at this link:
http://www.imdb.com/interfaces
For this project's purpose we used the plain-text form of the following tables:

- Actors
- Actresses
- Writers
- Directors
- Movies (~2 * 10^6)

IMDbPY

IMDbPY is a Python package useful to retrieve and manage the data of the IMDb movie database about movies, people, characters and companies.

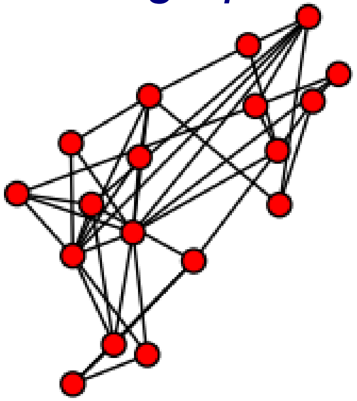
http://imdbpy.sourceforge.net/

We used the embedded python progran imdbpy2sql.py to convert the plain-text tables into an SQL database.
After that we write a python program that uses the IMDbPY libraries to query the previously created SQL database and outputs the arcs list in a plain-text file.
Nodes represents people (actors, actresses, writers and directors) while arcs represents a collaboration between nodes (i.e. two persons are connected by an arc if they worked on the same movie).

:::WebGraph

WebGraph is a framework for graph compression aimed at studying web graphs. It provides simple ways to manage very large graphs, exploiting modern compression techniques. With WebGraph you can access and analyse very large web graphs. This makes studying phenomena such as PageRank, distribution of graph properties of the web graph, etc. very easy.

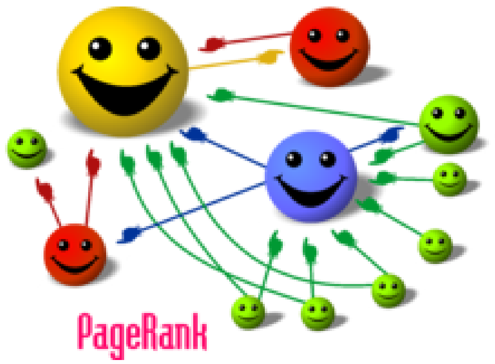
http://webgraph.dsi.unimi.it/



Starting from the previously generated arcs list (~3,3 GB), we used the WebGraph library to convert it into a compressed graph format (~278 MB). This graph format allows to manage a graph of huge dimension (~2,5 * 10^6 nodes and ~2,6 * 10^8 arcs) like our and calculate its properties.

PageRank

PageRank is a link analysis algorithm, used by the Google Internet search engine, that assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references.

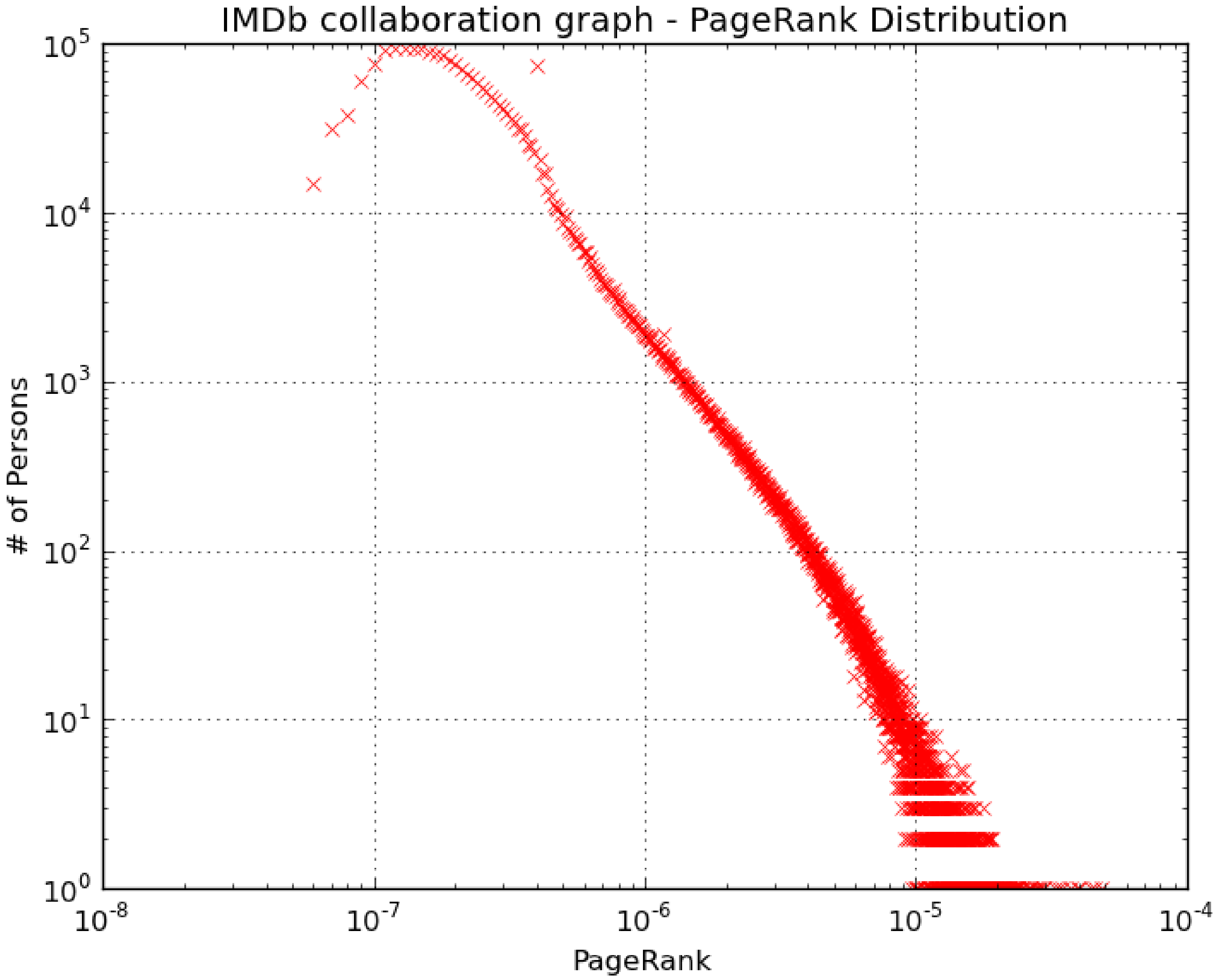


In the Webgraph library are implemented different kind of pagerank algorithm.
For our purpose we exploit the Power Method implementation. We ordered the result in ascending way. Here a sample:

PageRank	Node ID	Name
4.830324610696718E-5	1207	Jeremy, Ron
4.580297242927977E-5	1487	Shakespeare, William
4.431283436949331E-5	24348	Hitler, Adolf
4.037795185630405E-5	10248	Bush, George W.
3.9517780563064435E-5	6474	Reagan, Ronald
3.72614892710218E-5	26726	Pauw, Jeroen
3.4706558253899036E-5	67636	Kaufman, Lloyd
3.351618229652225E-5	3197	Clinton, Bill
3.2931214786648435E-5	3665	Wolf, Dick
2.9800095160876814E-5	23122	Goldberg, Whoopi

PageRank

Here the chart of the result returned by the Power Method PageRank algorithm. As we can see the curve follows the power law



This chart has been realised using the python library matplotlib

Degree Distribution

In the study of graphs and networks, the degree of a node in a network is the number of connections it has to other nodes and the degree distribution is the probability distribution of these degrees over the whole network.

We calculated the degree distribution starting from the arcs list in this way:

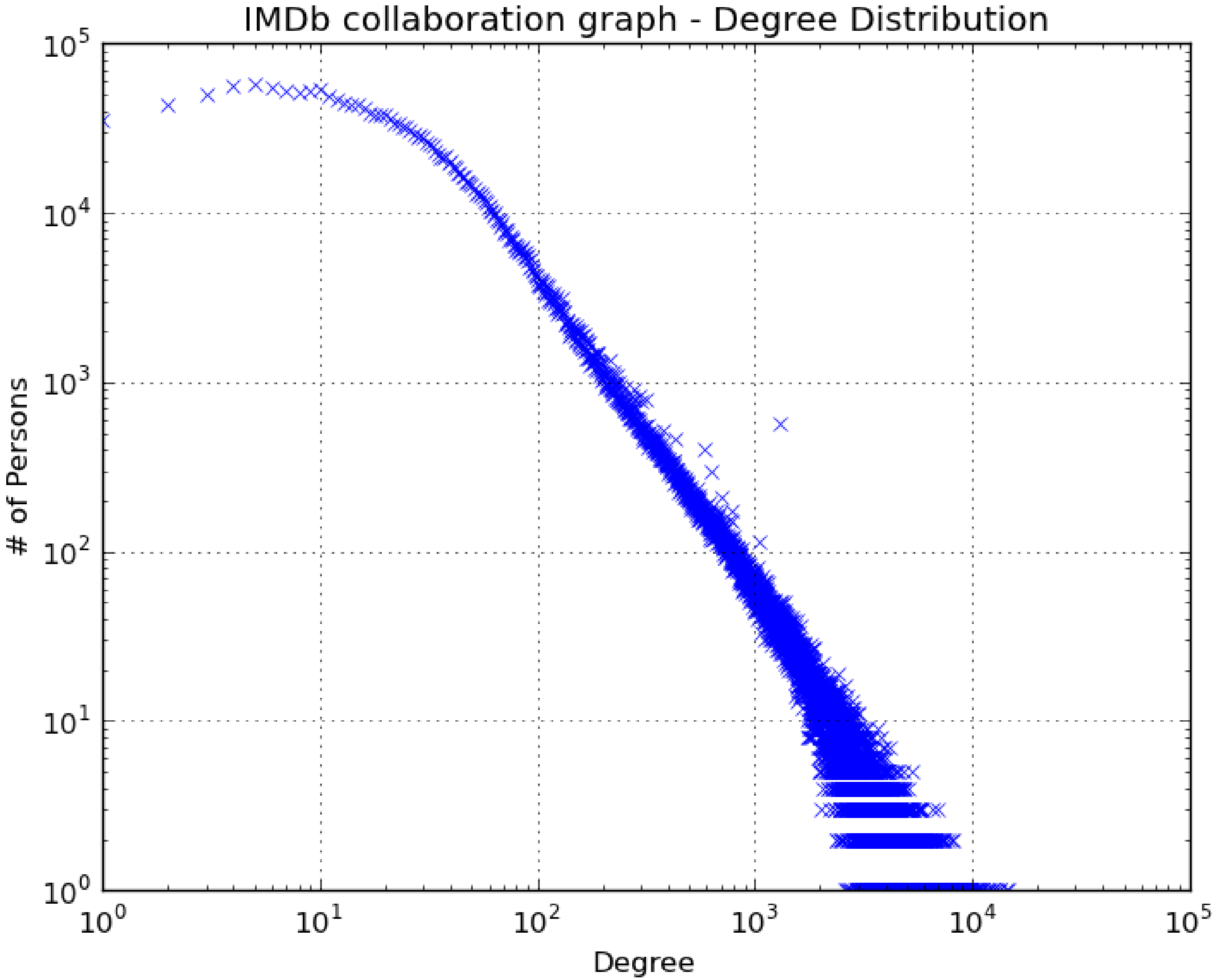
1. calculate the degree of every node
2. sort by increasing degree
3. for every degree return the number of nodes having that degree

With the following unix command:

```
awk '{print $1}' results/arcslists/imdbgrapharcslist | uniq -c  |
awk '{print $1}' | sort -n |  uniq -c | awk '{print $2, $1}'
> results/dd/imdbgraphdd.txt
```

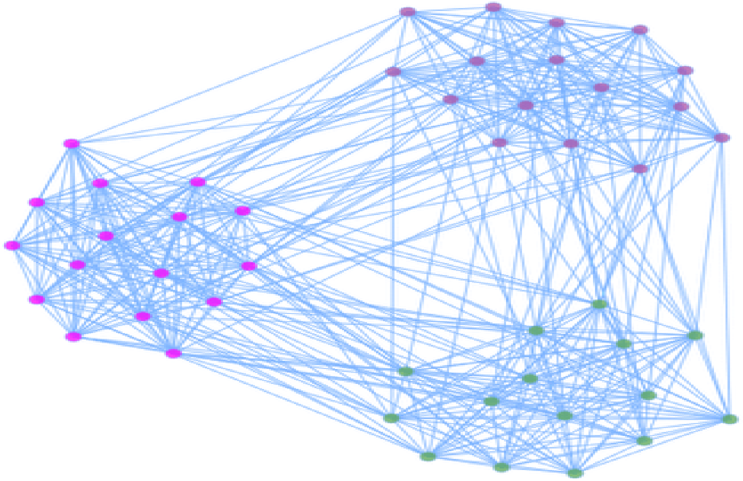
Degree Distribution

Here the chart of the Degree Distribution made with a Python script using the matplotlib library. As we can see the slope of the curve is similar to the pagerank's one.



Clustering with Metis

METIS is a set of serial programs for partitioning graphs, partitioning finite element meshes, and producing fill reducing orderings for sparse matrices. The algorithms implemented in METIS are based on the multilevel recursive-bisection, multilevel k-way, and multi-constraint partitioning schemes developed in our lab.



For computational reason we decided to run the Metis' clustering algorithm on a subgraph of 250000 nodes. In fact partitioning a graph of 2,5 millions of verteces in a “feasible” time requires a workstation.

We partion the graph in ten clusters and we run 5 iteration of the algorithm to refine the results.

Clustering Results

We ordered the nodes in every cluster by Page Rank. Then we used nodes with higher PageRank to characterize the cluster. What we obtained is that every cluster is characterized by mean of geographical areas where actors belongs or movie genre.
Here the results:

cluster0	Asia
cluster1	Adult
cluster2	Britain – Ireland – Netherland
cluster3	France – Italy – Spain – Portugal
cluster4	Germany – Switzerland – Austria
cluster5	Oldies
cluster6	Sportscaster
cluster7	Canada
cluster8	American vip (non professional actor)
cluster9	Hollywood

Assortativity

Assortativity is a preference for a network's nodes to attach to others that are similar. The assortativity coefficient r is the average value of nodes assortativities. Hence, positive values of r indicate a correlation between nodes of similar degree, while negative values indicate relationships between nodes of different degree. In general, r lies between −1 and 1.
For instance, in social networks, highly connected nodes tend to be connected with other high degree nodes. On the other hand, technological and biological networks typically show dissortativity, as high degree nodes tend to attach to low degree nodes. [Wikipedia]

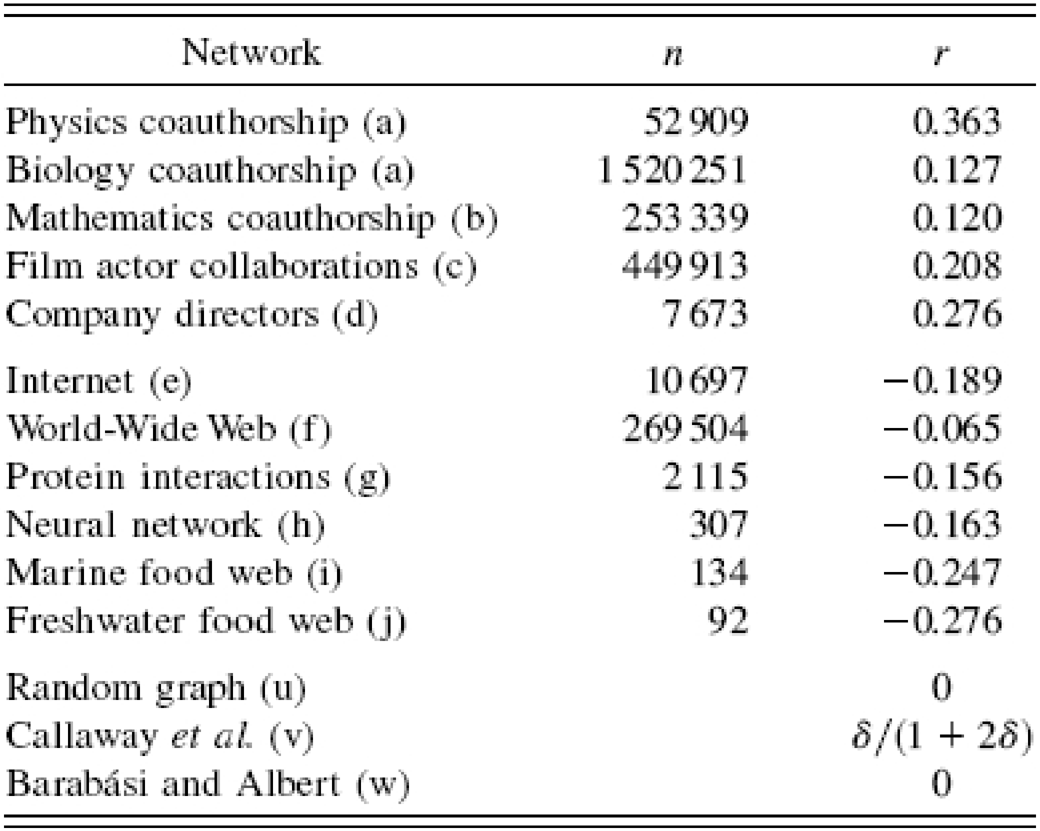
We calculated the Assortativity Coefficient for the graph using nodes PageRanks as similarity metric:

1. Calculate the assortativity for every node. The assortativity of node x is:
$\sum \text{neighbor}(x).\text{rank} / \sum \text{neighbors}(x)$

2. Calculate the assortativity coefficient of the graph:
$\sum \text{node.assortativity} / \sum \text{node}$

Assortativity Results

The Assortativity Coefficient that we have found for our graph is ~0,35. This result is in line with the values of social networks graphs that have positive assortativity coefficient as it is shown in the following table:



Network	n	r
Physics coauthorship (a)	52 909	0.363
Biology coauthorship (a)	1 520 251	0.127
Mathematics coauthorship (b)	253 339	0.120
Film actor collaborations (c)	449 913	0.208
Company directors (d)	7 673	0.276
Internet (e)	10 697	−0.189
World-Wide Web (f)	269 504	−0.065
Protein interactions (g)	2 115	−0.156
Neural network (h)	307	−0.163
Marine food web (i)	134	−0.247
Freshwater food web (j)	92	−0.276
Random graph (u)		0
Callaway *et al.* (v)		$\delta/(1+2\delta)$
Barabási and Albert (w)		0

Assortativity Results

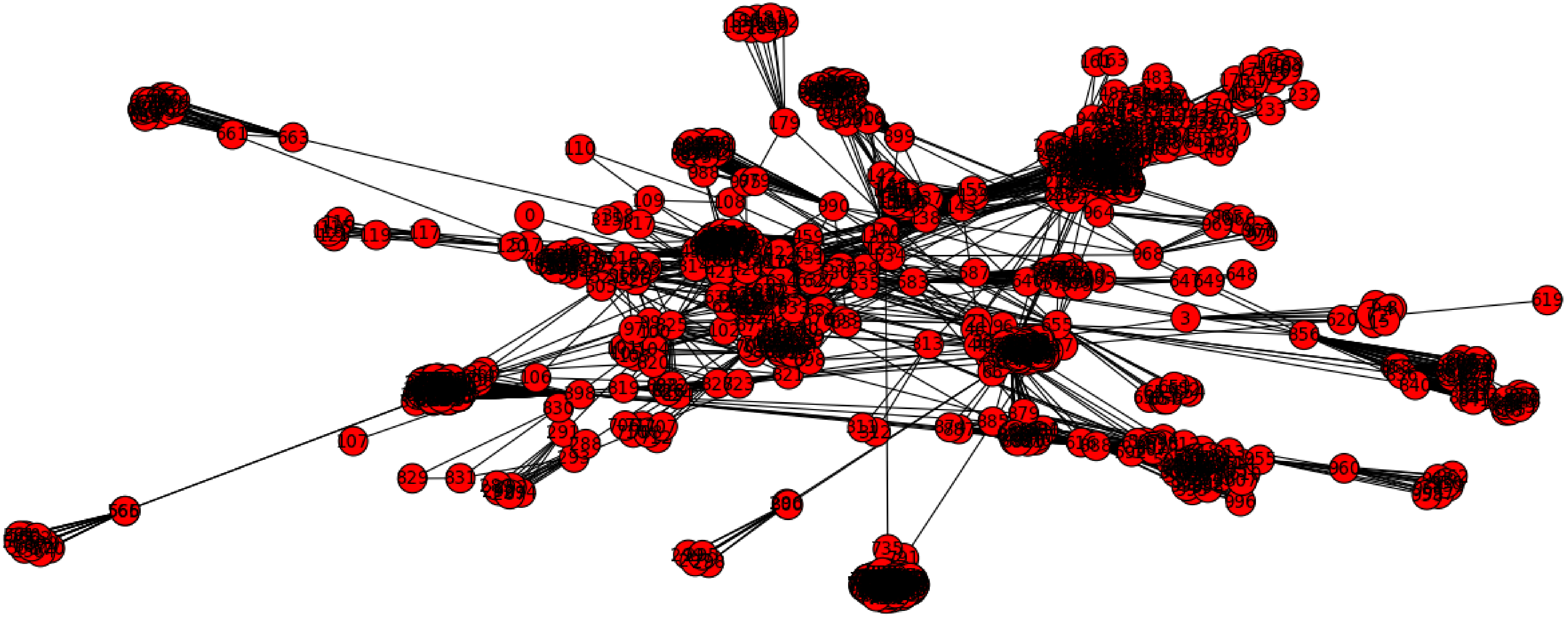
So, in general, social networks appear assortative while technological and biological networks appear to be disassortative.

> *“This might happen because most networks have a tendency to evolve, unless otherwise constrained, towards their maximum entropy state which is usually disassortative. [Wikipedia]”*

In our case, as the clustering partition suggests, constraints to the evolution of the network could be given by space and time. The space constraint is due to the fact that people living in the same geographical area have an higher probability to make a movie together, while the time constraint is due to limitation of human being life. Obviously the time constraint is less evident considering that the movies industry lasts since few generation.

Graph Visualization

To give an idea of the shape of the graph we draw a part of it using NetworkX library for Python (http://networkx.lanl.gov/)



661
663
110
179
899
109
108
990
0
116
119
117
155
145
138
161
163
483
232
233
964
968
641
649
648
687
683
3
619
620
856
840
655
813
106
898
819
821
879
311
312
960
996
107
566
830
829
831
291
288
253
886
735